

CLUSTER FINDING

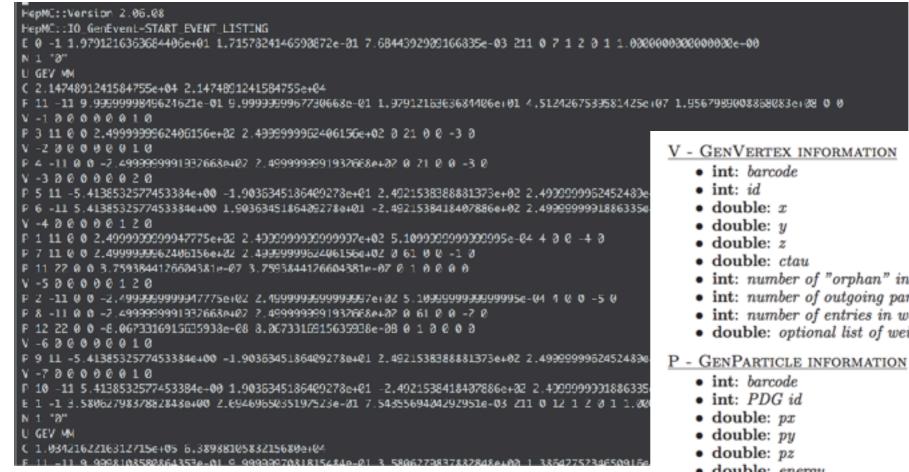
MATT ZHANG



Cluster Finding

- Problem recap given an event with multiple objects (such as two jets), we want to be able to "zoom in" and cut out a section of the ECAL and HCAL around each object.
- Three methods of doing this:
 - Pass truth 4-vectors to python script for creating ECAL and HCAL sections.
 - Use PandoraPFA package to find and return clusters from ROOT files, then use python script to cut out ECAL and HCAL sections.
 - Modify python script to directly find clusters in ROOT files, as opposed to just calculating the barycenter.

Truth 4-Vectors



- int: number of "orphan" incoming particles
- int: number of outgoing particles
- int: number of entries in weight list (may be zero)
- double: optional list of weights

- double: energy
- double: generated mass
- int: status code
- double: Polarization theta
- double: Polarization phi
- int: barcode for vertex that has this particle as an incoming particle
- int: number of entries in flow list (may be zero)
- int, int: optional code_index and code for each entry in the flow list

Truth 4-Vectors

- Need to decide what to take as our truth 4-vectors here. Should we take the first two particles with PdgID corresponding to π0 and use their (px, py, pz, m) vector?
- Working on a script to take a HepMC file and extract the relevant 4-vectors in a text file (after which the file can be used by the python script).
- Need to know about detector geometry given an eta and phi, at what (x, y, z) will the calorimeter be hit?
- Deciding which vectors to take as truth 4-vector is process dependent. This should not be a big deal if we are only generating a few types of processes.

Pandora

- Another option is to use the DDMarlinPandora package.
- This can take the output of Marlin and send it to PandoraPFA, which outputs things like cluster positions.
- DDMarlinPandora is an offshoot of the MarlinPandora package and appears to be in development - it may be a bit difficult to use right now.

Custom Algorithm

- We have the entire ECAL and HCAL information. Based on this we can create a simple clustering algorithm in python.
- Start with a grid of values \rightarrow find the pixel with the highest value.
 - Look in a radius of two pixels around that pixel, and for all pixels which have a lesser value, but still above the threshold, add it to the cluster.
 - Continue doing this until the cluster no longer grows.
 - Repeat process with highest-value remaining pixel.
- This approach would allow us to get variable-sized chunks of ECAL and HCAL data (containing the entire cluster), rather than a uniform sized slice.